Please amend the claims as follows:

1. (currently amended) A software object that contains symbolic names and is executable in an execution environment for the software object, the execution environment resolving the symbolic names, the symbolic names including system symbolic names defined in the execution environment, the execution environment executing in a processor, and the software object being contained in a memory device accessible to the processor, and

the software object comprising:

one or more obfuscated symbolic names that correspond to system symbolic names;

a first association between the obfuscated symbolic names and encrypted forms of the corresponding system symbolic names; and

a static watermark that has been added to the ~~code~~software object,

the execution environment including a second association of the encrypted forms with information needed to resolve the corresponding system symbolic names and the execution environment using the first and second associations to resolve the obfuscated symbolic names, and using the static watermark to determine whether the software object has been altered prior to the software object being executed in the program execution environment.

2. (previously presented) The software object set forth in claim 1 wherein:

the static watermark's value is a digest of the software object prior to addition of the static watermark.

3. (previously presented) The software object set forth in claim 1wherein the software object further comprises:

other obfuscated names that replace names defined in source code from which the software object was made.

4. (previously presented) The software object set forth in claim 1 wherein:

the software object is downloaded to the program execution environment for execution.

5. (previously presented) The software object set forth in claim 1, the software object further comprising;

an encrypted first key, the first key having been used to produce the encrypted forms of the corresponding system symbolic names,

the execution environment having access to a second key that can decrypt the first key; and

the execution environment using the second key to decrypt the first key and the first key to make the encrypted forms in the second association.

6. (Original) An improved class loader that loads a class in a program execution environment in a host computer system, the class being required for execution of a program in the program execution environment and the program including a first association between symbolic names in the program and encrypted forms of symbolic names defined in the class and the improved class loader being characterized in that:

the improved class loader extends the class on execution of the program in the program execution environment by

using the first association and a second association between the encrypted forms and information used to resolve the symbolic names defined in the class to resolve the symbolic names in the program, and

adding a method to the program which determines whether the program has been modified by the host.

7. (Original) The improved class loader set forth in claim 6 wherein:

the method is encrypted prior to being added to the program; and

the improved class loader decrypts the method on adding the method to the program.

8. (Original) The improved class loader set forth in claim 7 wherein:

the program includes information from which the method determines whether the program has been modified by the host.

9. (Original) The improved class loader set forth in claim 6 wherein:

the program includes a static watermark; and

the static watermark is the information from which the method determines whether the program has been modified by the host.

10. (Original) The improved class loader set forth in claim 9 wherein:

the static watermark's value is a digest of the program prior to addition of the static watermark.

11. (Original) The improved class loader set forth in claim 9 wherein:

The static watermark is at a location in the program that is determined by a key; and

the method has access to the key and uses the key to locate the static watermark.

12. (Original) The improved class loader set forth in claim 6 wherein:

the improved class loader has access to an encryption key that was used to produce the encrypted forms in the first association; and

the improved class loader uses the encryption key to produce the second association on loading the class.

13. (Original) The improved class loader set forth in claim 12 wherein:

the program includes an encrypted form of the encryption key used to produce the second association; and

the improved class loader obtains the encryption key by using a decryption key to decrypt the encrypted form of the encryption key.

14. (Original) A method of protecting a program that is executed in a host computer system from the host, the program being executed in a program execution environment that loads a class, the class being required for execution of the program in the program execution environment, and the program including a first association between symbolic names in the program and encrypted forms of symbolic names defined in the class,

the method being characterized by the steps performed in the class loader of:

making a second association between the encrypted forms and information used to resolve the symbolic names defined in the class, the first and second associations being used to resolve the symbolic names, and

adding a method to the program which determines whether the program has been modified by the host.

15. (Original) The method set forth in claim 14 wherein:

the added method is encrypted; and

the step of adding the method includes the step of decrypting the method.

16. (Original) The method set forth in claim 14 wherein:

the program includes information which the added method uses to determine whether the program has been modified by the host.

17. (Original) The method set forth in claim 14 wherein:

the program includes a static watermark; and

the static watermark is the information used by the added method to determine whether the program has been modified by the host.

18. (Original) The method set forth in claim 17 wherein:

the static watermark's value is a digest of the program prior to addition of the static watermark; and

the added method reads the static watermark, recomputes the digest, and compares the recomputed digest with the watermark's value.

19. (Original) The method set forth in claim 17 wherein:

the added method uses a key to locate the static watermark in the program.

20. (Original) The method set forth in claim 14 wherein:

the class loader has access to an encryption key that was used to produce the encrypted forms in the first association; and

the step of making a second association includes the steps of accessing the encryption key and using the encryption key to produce the encrypted forms.

21. (Original) The method set forth in claim 20 wherein:

the program includes an encrypted form of the encryption key that was used to produce the encrypted forms; and

the step of accessing the encryption key includes the step of using a decryption key to decrypt the encrypted form of the encryption key.

22. (Previously presented) A method of protecting a software object that is executed in a host computer system from the host, the software object being executed in an execution environment for the software object in the host computer system, the execution environment loading a class that is used in executing the software object in the execution environment and the method being characterized by:

steps performed prior to execution of the software object in the execution environment comprising

replacing symbolic names in the software object that are defined in the class with obfuscated symbolic names corresponding thereto; and

making a first association between the obfuscated symbolic names and encrypted forms of the replaced symbolic names; and

steps performed during the execution of the software object in the execution environment comprising

making a second association between the encrypted forms of the symbolic names and information required to resolve the symbolic names;

adding a method to the software object that determines whether the software object has been modified by the host;

using the first and second associations to resolve the obfuscated symbolic names; and

executing the added method to determine whether the software object has been modified by the host.

23. (Previously presented) The method of protecting the software object set forth in claim 22 further characterized in that:

the steps performed prior to executing the software object further comprise the step of

obfuscating other symbolic names that are not defined in the class.

24. (Previously presented) The method of protecting the software object set forth in claim 22 further characterized in that:

the method to be added is encrypted; and

the step of adding the method includes the step of decrypting the method.

25. (Previously presented) The method of protecting the software object set forth in claim 22 further characterized in that:

the software object includes information from which the method can determine whether the software object has been modified by the host; and

in the step of executing the added method, the added method uses the information to determine whether the software object has been modified by the host.

26. (Previously presented) The method of protecting the software object set forth in claim 25 further characterized in that:

the steps performed prior to executing the software object further comprise

adding a static watermark to the software object; and

the static watermark is the information used by the added method.

27. (Previously presented) The method of protecting the software object set forth in claim 26 further characterized in that:

in the step of adding the static watermark, the location of the static watermark in the software object is determined by a key; and

in the step of executing the added method, the added method uses the key to locate the watermark.

28. (Previously presented) The method of protecting the software object set forth in claim 22 further characterized in that:

the step of making the second association includes the steps of

obtaining a key used to make the encrypted forms in the first association and

using the obtained key to make the encrypted forms in the second association.

29. (Previously presented) The method set forth in claim 28 further characterized in that:

the software object includes an encrypted form of the encryption key that was used to make the encrypted forms in the first association; and

the step of obtaining the key includes the step of using a decryption key to decrypt the encrypted form of the encryption key.